Realising an Applied Gaming Eco-sytem

Research and Innovation Action

Grant agreement no.: 644187

D2.2 - Final Bundle of Server-side Components (WP2)

RAGE – WP2 – D2.2

Project Number	H2020-ICT-2014-1
Due Date	31 July 2017
Actual Date	13 July 2017
Document Author/s	Antonio Calvo Morata, Baltasar Fernández Manjón, Manuel Freire Morán, Dan Cristian Rotaru, Iván Martínez Ortiz, Ruben Riestra, Mihai Dascalu, Raja Lala, Wim Westera
Version	1.2
Dissemination level	PU
Status	Final
Document approved by	BFM

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644187

Document Version Control			
Version	**Date**	**Change Made (and if appropriate reason for change)**	**Initials of Commentator(s) or Author(s)**
0.1	13 July 2017	First version	ACM, DCR, MFM, IMO, BFM
1.0	21 July 2017	Second complete version	ACM, DCR, MFM, IMO, BFM, RR, MD, RL
1.2	31 July 2017	Reviewed version with minor changes	ACM, DCR, MFM, IMO, BFM, RR, MD, RL, WW

Document Change Commentator or Author		
Author Initials	**Name of Author**	**Institution**
ACM	Antonio Calvo Morata	UCM
BFM	Baltasar Fernández Manjón	UCM
MFM	Manuel Freire Morán	UCM
DCR	Dan Cristian Rotaru	UCM
IMO	Iván Martínez Ortiz	UCM
RR	Ruben Riestra	Inmark
MD	Mihai Dascalu	UPB
RL	Raja Lala	UU
WW	Wim Westera	OUNL

Document Quality Control			
Version QA	**Date**	**Comments (and if appropriate reason for change)**	**Initials of QA Person**
N/A			

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

UCM	Universidad Complutense de Madrid
UPB	University Politehnica of Bucharest
UU	Utrecht University
OUNL	Open University of the Netherlands

EXECUTIVE SUMMARY

This document describes the final bundle of server side components, including a description of their functionality, and links to their full designs, documentation and downloadable versions. This bundle aggregates only the WP2 assets. Other server-side assets not described here will be covered in the final WP3 deliverables (e.g. SUGAR engine).

This document links the asset catalogue with the more technical description of the server assets produced in the project.

For every asset, the following items are available:

1. Catalogue product fact sheet for the asset or asset suite/bundle. To simplify asset understanding and adoption by the game industry, fact sheets may aggregate different individual assets descriptions into a single sheet.
2. The final and updated asset information and links to the source code and documentation.
3. The source code with complete and updated documentation in an open repository (such as a GitHub wiki or similar).

All the updated asset information is always available on-line at the public repositories where the assets are stored. Notice that those assets as open software will be continuously improved by their creators till the end of the project (the project has decided to extend that WP task till month 48) and beyond it.

This document is focus on server-side components of the WP2, for a full description of the related client-side components of this workpackage, please refer to D2.4 - Final Bundle of Client-side Components (WP2).

1 GAS: GAMING ANALYTICS SUITE

The Gaming Analytics Suite (GAS) comprises the main information of the 5 UCM Assets:

1. Server-Side Interaction Storage and Analytics
2. Authentication & Authorization
3. Game Storage, Server-Side
4. Server-side Dashboard and Analysis
5. Client Tracker

Table 1. Gaming Analytics Suite factsheet

	**GAS** **Gaming Analytics Suite** **UCM Assets:** **Server-Side Interaction Storage and Analytics** **Authentication & Authorization** **Game Storage, Server-Side** **Server-side Dashboard and Analysis** **Client Tracker**
Short description	Provides data and insights on game usage performance
Type of product	Integrated pack of game components
Main benefit for Game Studio	• **Gain creator**: Evidence based enhanced games: enables games ROI calculation, thus boosting game saleability • **Pain reliever**: improved feedback and intelligence, cutting time to market and improving game´s fit for purpose
Main features for Game **designers/developers**	• GAS covers the entire process needed to create, review and communicate game´s empirical performance evidence. GAS enables the configuration and execution of data gathering from in-game interactions, their analysis, querying, and dashboard visualization • Enriched data-evidence produced by GAS is useful both as decision making support among users (buy-in critical success factor), as well as studio´s internal monitoring of game's core logic, mechanics and performance, allowing better and quicker improvements.
Main features for **Game customer/user**	Allows monitoring and evaluation of end users´ performance and progress towards pursued objectives.
Creator	Complutense University of Madrid, UCM
Further info	https://github.com/e-ucm/rage-analytics/wiki

In this document we describe the **server-side assets**.

1.1 Included assets

1.1.1 Server-Side Interaction Storage and Analytics

This asset offers a ready-to-deploy server-side implementation of a data collection and storage service. It provides an API to manage games and classes, and, also, the data collected by the tracker. The data stored by the analysis can be shown via Server-side Dashboard and Analysis Asset.

Complete asset specification and analysis configuration was previously described in D2.5.

New analyses have been developed for games using the xAPI serious games model (http://xapi.e-ucm.es/vocab/seriousgames).

Links

- Sources:
  https://github.com/e-ucm/rage-analytics-backend
- Release:
  https://github.com/e-ucm/rage-analytics-backend/releases/latest
- Design:
  https://docs.google.com/document/d/1imtLyI59yOv9CrPRZdSu8UaUmGs_3ZyxhQYphcmnUPE/edit?usp=sharing
- User documentation:
  https://github.com/e-ucm/rage-analytics-backend/wiki
- Complete RAGE Analytics documentation:
  https://github.com/e-ucm/rage-analytics/wiki
- Launcher:
  https://github.com/e-ucm/rage-analytics
- REST API
  http://e-ucm.github.io/rage-analytics-backend/

1.1.2 Authentication & Authorization

This asset provides a central location where client-side assets can authenticate and locate server-side assets, including analytics. Servers can also register to locate other servers and be locatable by clients, restricting access using configurable roles.

Asset functionality was previously described in D2.5. Complete diagram of the asset in the architecture can be seen in Figure 1.

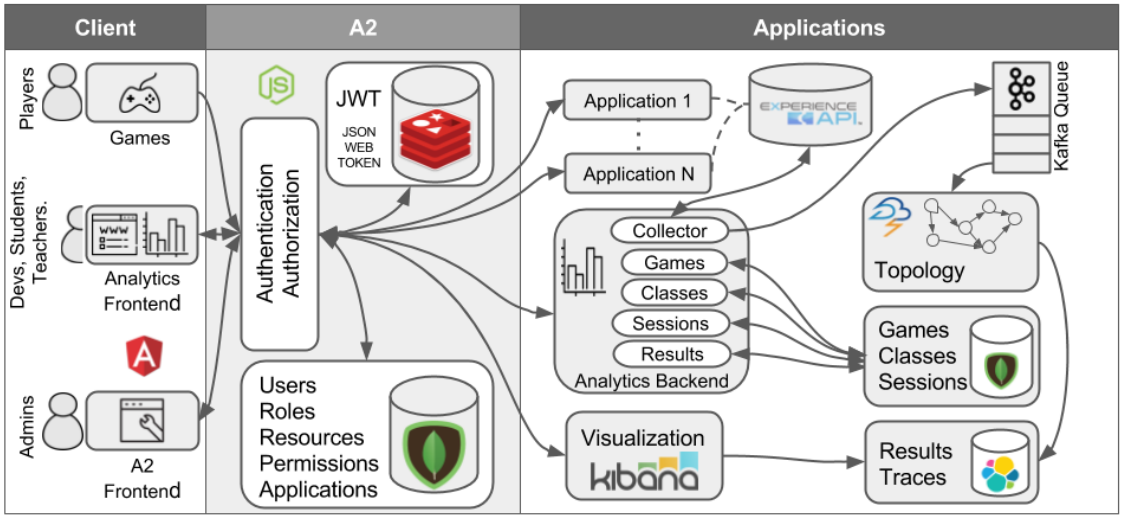

Figure 1. Authentication & Authorization in RAGE Analytics.

Links

- Sources:
  https://github.com/e-ucm/a2
- Release:
  https://github.com/e-ucm/a2/releases/latest
- Design:
  https://docs.google.com/document/d/1Ve6qGs30uGUYrlMBsB-UrlMaaFw2VdbSKROml8_TT78/edit?usp=sharing
- User documentation:
  https://github.com/e-ucm/a2/wiki
- Complete RAGE Analytics documentation:
  https://github.com/e-ucm/rage-analytics/wiki
- Launcher:
  https://github.com/e-ucm/rage-auth2
- REST API documentation:
  http://e-ucm.github.io/a2/

1.1.3 Server-Side Dashboard and Analysis

This asset provides the user interface to access the analytics functionalities of the system. The stakeholders can log in using different roles to access different functionalities. The available roles are: *student*, *teacher,* and *developer.*

As previously described in D2.5, Kibana is used due to its provided features. Default analyses and visualizations based on the xAPI model developed with ADL Co-Lab are now available and they have been tested with a demo game (Serrano, 2017).

Pluggable custom analyses and visualizations can be added according the requirements of the clients or games (Freire, 2016). This allows for game-specific analyses and visualizations (Alonso, 2017), as depicted in Figure 2.

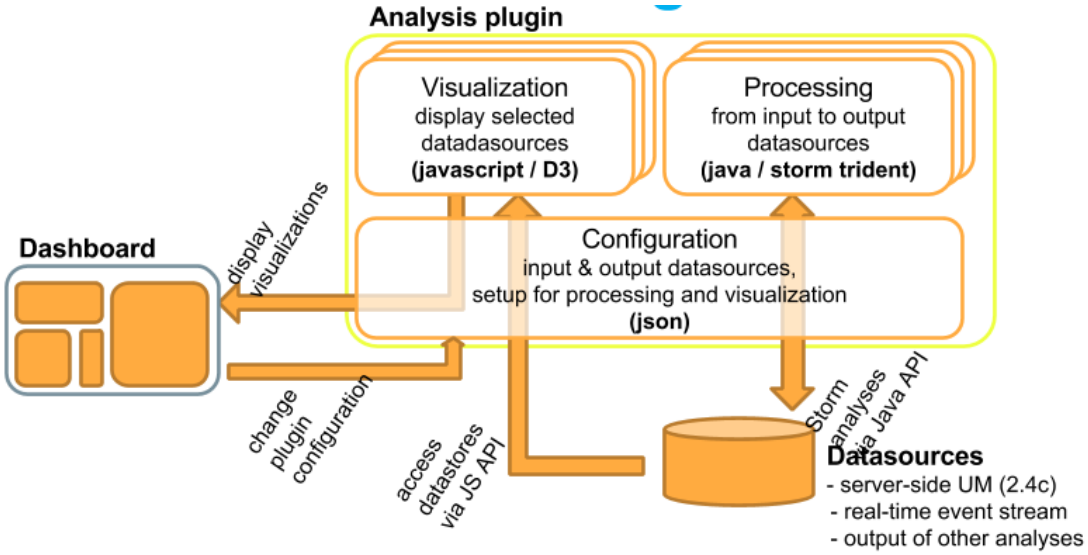

Figure 2. Configuration of analyses and visualizations.

Default visualizations have been developed for both *teachers* and *developers*. For instance, Figure *3* shows some of the default visualizations available for *developers*.

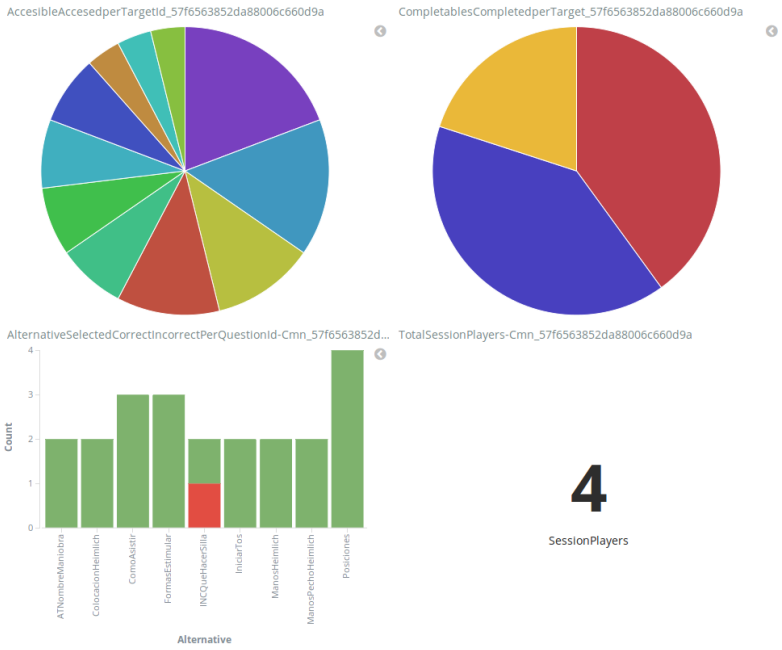

Figure 3. Some of the default visualizations.

Links

- Sources:
  https://github.com/e-ucm/rage-analytics-frontend
- Release:
  https://github.com/e-ucm/rage-analytics-frontend/releases/latest
- Design:
  https://docs.google.com/document/d/13rrTugiezXwNTfXriQeg1l-jb0H4Jil7l4inpXCVvm8/edit?usp=sharing

- User documentation:
  https://github.com/e-ucm/rage-analytics-frontend/wiki
- Complete RAGE Analytics documentation:
  https://github.com/e-ucm/rage-analytics/wiki
- Launcher:
  https://github.com/e-ucm/rage-analytics

1.1.4 Game Storage Server-Side

This asset allows games (or indeed any authenticated client or server) to store game data in the server, associated to arbitrary keys, and later retrieve, modify or erase this data.

This is the server counterpart to the Client-side Game Storage Asset, which provides a simple client side library to store, modify, retrieve and delete the data managed by this asset.

Links

- Sources:
  https://github.com/e-ucm/rage-gamestorage-server
- Release:
  https://github.com/e-ucm/rage-gamestorage-server/releases/latest
- Design:
  https://docs.google.com/document/d/1j2Mb7k8PEAJeK5e-j5qXtMs2-JzWWFNMR__C9NStG_k/edit
- User documentation:
  https://github.com/e-ucm/rage-analytics/wiki/Game-Storage-Server
- Complete RAGE Analytics documentation:
  https://github.com/e-ucm/rage-analytics/wiki
- Launcher:
  https://github.com/e-ucm/rage-analytics
- REST API documentation:
  http://e-ucm.github.io/rage-gamestorage-server/

1.1.5 Client Tracker

This component sends analytics information to a server; or, if the server is currently unavailable, stores them locally until it becomes available again (this is a semi-connected mode). The tracker implementations were updated as previously described in D2.5 to suit the xAPI model for serious games (accessible at: http://xapi.e-ucm.es/vocab/seriousgames), described in (Serrano, 2017).

Complete description of this asset can be found in D2.4 - Final Bundle of Client-side Components (WP2).

2 EASY DIALOGUE INTEGRATOR

Table 2. Easy Dialogue factsheet

	**EDI** **Easy Dialogue Integrator** **UU Assets:** **Step based competence assessment** **Communication Scenario Editor**
Short description	End-user-friendly dialogue authoring tool and reasoner for seamlessly incorporating expressive dialogues between virtual character(s) and a player
Type of product	Bundled dialogue creation tools
Main benefit for Game Studio	**Gain creator.** Powerful value proposition: Easy to use, customer centric designed, allows domain-expert users to create expressive ad-hoc (customised) dialogue scenarios **Pain Relievers:** the expert-user created dialogue is seamlessly integrated in a game, thus reducing costs for expanding usage base
Main features for Game **designers/developers**	• Tested and robust after 3 years of usage in different communication-skills learning environments. • Extensive documentation, game developers from RAGE installed and integrated in their games self-sufficiently. • Easy adaptation to multiple scenarios by empowering programmers to integrate dialogues seamlessly in a game.
Main features for **Game customer/user**	Easy to use, even for non-programmers. EDI provides the features from most dialogue tools and in addition expressive dialogue constructs. Communication-skills experts develop and modify a scenario independent of a programmer. Runs in a web browser as opposed to existing client-based tools, making EDI more easily accessible to non-professional authors.
Creator	Utrecht University (UU)
Further info	https://github.com/UURAGE https://uudsl.github.io/scenario https://github.com/UURAGE/ScenarioEditor/tree/master/doc https://github.com/UURAGE/ScenarioReasoner/tree/master/doc https://youtu.be/rY7mj2my5MU https://youtu.be/-MW8j5EDL6Y

In this document only the **server-side asset only, i.e. Step based competence assessment,** is described. For the client-side please refer to D2.4 deliverable.

2.1 Included assets

2.1.1 Step-based competency assessment asset

This asset is typically used in a game in conjunction with the communication scenario editor. While the editor makes it easy for a non-programming expert to develop dialogue content, the step-based competency assessment asset makes it simple for a programmer to integrate dialogues seamlessly in a game.
The step-based competency assessment asset handles the expressiveness of the dialogue constructs of a scenario produced by the communication scenario editor.

The asset has two main functions for a game programmer: a scenario parser and a scenario reasoner. The former parses a scenario produced using the scenario editor, and returns an ID of the scenario. A game at run-time interacts with the scenario reasoner using the ID of the parsed scenario. The reasoner then provides information about the possible following steps at each step in the series of interactions. The scenario reasoner also returns parameter values (e.g. incremental scores) and property values (e.g. emotional effects on an NPC) to a game.

When should a game developer use the asset?
Use this in conjunction with the communication scenario editor asset. The relation between the two assets is shown schematically below:

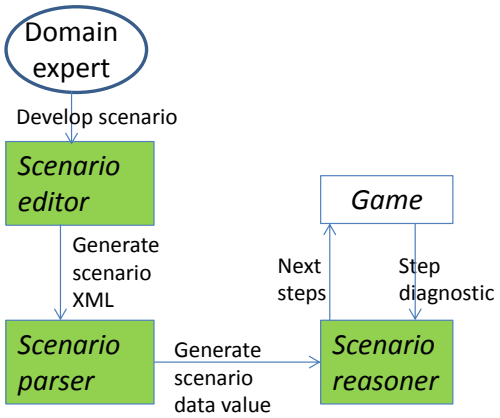

Figure 4. Scenario editor to simplify dialogue creation

Use the scenario editor when you want domain-/content- experts/teachers to develop (a lot of different) scenarios for your game. Use the step-based competency assessment asset to seamlessly integrate these dialogues into your game.

We have designed the editor and the assessment assets to be loosely coupled using a REST architecture. The editor produces an xml that follows an instance of the schema: https://uudsl.github.io/scenario

The reasoner asset processes any valid scenario produced according to this schema, using any valid configuration (single VC / multiple VC's; configurable scores, parameters, properties etc).

How to use the Step-based assessment asset
A scenario that follows the schema can be parsed by the method ParseScenario, which is for a one-time conversion from XML (output from the editor) to a binary representation. This representation need not change unless the XML changes (i.e. there is a changed scenario/dialogue). This method returns the ID of the binary that are needed for the other API calls.

The scenario reasoned is a web-service that offers its services through JSON-RPC. A game interacts at run-time with the scenario reasoner (with the ID of the specific parsed scenario).

The reasoner has the following methods:

- ScenarioInfo: returns info about the scenario like metadata, definitions and properties of the scenario and the characters.
- Examples: returns an initial state that contains the initial values for the parameters that persist throughout the playthrough like the initial emotion of a character.
- AllFirsts: takes a state from the list of states returned by this method or the initial state. The list of states returned contains a state per next step (a node in the editor). A step has a type (player, computer or situation) and can be handled accordingly. A step also contains the statement text and the property values and prospected parameter values after that step has been done. If the returned list is empty the dialogue scenario has ended.

The JSON input and output of the methods listed above are detailed in the following: https://github.com/UURAGE/ScenarioReasoner/tree/master/doc/schemas

The two assets Communication Scenario Editor (WP3) & Step-based assessment are typically used together in a game. It is of course possible to use the XML output from the editor; however in that case the game-developer needs to develop their own “parser” and “reasoner” modules.

Technical and quality aspects
The asset is implemented in Haskell on top of the Ideas framework from Utrecht University. The Ideas framework has been tested extensively both functionally and non-functionally and used in diverse domain reasoners (DME secondary math education, Math-Bridge, MathDox, Logic tool, Ask-Elle, tutor for Haskell) in addition to the step-based assessment asset.

Links

- Sources:
  https://github.com/UURAGE
- Release:
  https://github.com/UURAGE/ScenarioReasoner
- Design:
  https://github.com/UURAGE/ScenarioReasoner/blob/master/doc/Software%20design%20document.pdf
- User documentation:
  https://github.com/UURAGE/ScenarioReasoner/tree/master/doc
- Launcher:
  https://github.com/UURAGE/ScenarioReasoner/blob/master/ScenarioReasoner.cabal
- REST API
  https://uudsl.github.io/scenario

3 READERBENCH ADVANCED NATURAL LANGUAGE PROCESSING FRAMEWORK

Table 3. ReaderBench, Advanced Natural Language Processing Pack factsheet

	**ReaderBench Advanced Natural Language Processing Pack** **UPB Assets:** **ReaderBench – Sentiment Analysis on Texts**
Short description	Advanced Multi-Lingual Open-Source Framework that extracts value from texts using Natural Language Processing
Type of product	Bundled game components
Main benefit for Game Studio	**Gain creator**: Empowerment of games by enhancing identification and understanding of players´ behaviours expressed through text inputs their sentiments, the relevant topics outlined in discussions that take place between players, the textual complexity of their written statements, their degree of participation in collaborative groups or their comprehension potential derived from the textual traces. **Pain reliever**: Improves the personalisation of the game according to the textual complexity and other text-related features extracted from players' written contributions.
Main features for Game **designers/developers**	Using ReaderBench, you will be able to: • Identify and rank sentiments (excited, sad, scared, angry, tender and happy) expressed by the players' textual productions and personalize the game accordingly. • Extract and rank the most relevant keywords expressing the topics covered by the players in their game discussions. • Perform multi-dimensional analysis of textual complexity factors reflecting relevant writing style information with regards to the players' profile. • Discover the different degrees of activity, involvement, and interaction of participants within the discussion in a game. • Determine specific reading strategies used by players, useful for reliable prediction of the player's comprehension with regards to a target text.
Main features for	Users, e.g. educationalists, will be able to:

Game customer/user	• Gather the sentiments expressed by students regarding a course or some specific areas within it. • Assess the main keywords and/or topics covered within students' assignments' or in their discussions. • Generate a list of the most relevant keywords covered within the course resources and adapt them accordingly to required topics. • Automatically assess the quality of text assignments produced by students through the use of a wide range of textual complexity indices. • Analyze the behavior and involvement of participants in a collaborative discussion platform centered on problem-solving tasks or creativity stimulation. • Estimate the comprehension level of learners based on essays, self-explanations and summaries in order to adapt future reading materials.
Creator	UPB
Further info	http://readerbench.com/docs/api

3.1 Included assets

3.1.1 ReaderBench – Sentiment Analysis on Texts

The **Sentiment Analysis on Texts** component determines the strongest sentiment expressed by a text. Given a text, the component provides relevance scores for six major sentiments (excited, sad, scared, angry, tender and happy) ordered descending by relevance. Relevance scores are provided for each sentiment in terms of strength express as percentage.
This component processes texts written in English and French languages.

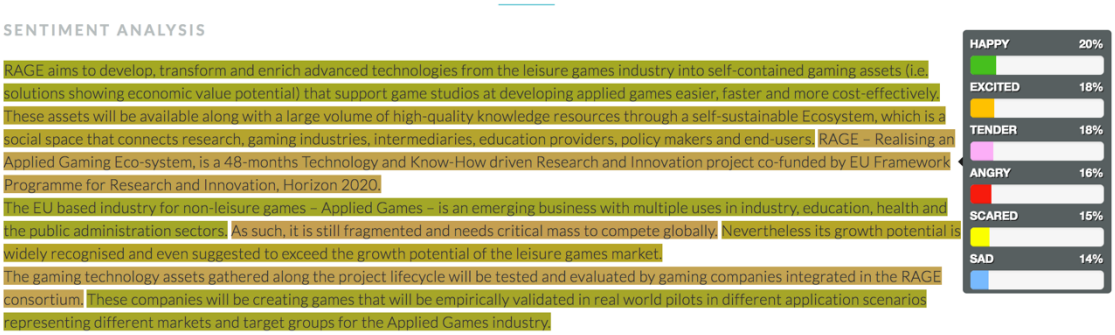

Figure 5. Example of Sentiment Analysis response using the online demo client

Links

- Sources: https://git.readerbench.com/ReaderBench/ReaderBench
- Release: https://git.readerbench.com/ReaderBench/ReaderBench/tags
- Design: http://readerbench.com/docs/sentiment-analysis/sdd
- User documentation:

http://readerbench.com/docs/sentiment-analysis/manual

- Complete RAGE ReaderBench b documentation: https://git.readerbench.com/ReaderBench/ReaderBench/wikis/home
- Launcher: http://readerbench.com/deployment
- REST API http://readerbench.com/docs/api

4 CONCLUSIONS

This deliverable describes the final catalogue of WP2 server-side assets produced by RAGE. All of them are fully functional and include a complete documentation for promoting their adoption by the game industry.

Notice that open software is in continuous evolution for corrective maintenance and to respond to clients' new requirements or needs. These changes will be reflected in the on-line documentation of the assets. Therefore, those assets will be continuously improved by their creators till the end of the project (the project has decided to extend that WP task till month 48) and beyond it.

REFERENCES

(Alonso, 2017) Alonso-Fernández, C., Calvo Morata, A., Freire, M., Martínez-Ortiz, I., Fernandez-Manjon, B.: ***Systematizing game learning analytics for serious games***. In: IEEE Global Engineering Education Conference (EDUCON). pp. 1106–1113 (2017).

(Freire, 2016) Freire, M., Serrano-Laguna, Á., Iglesias, B.M., Martínez-Ortiz, I., Moreno-Ger, P., Fernández-Manjón, B.: ***Game Learning Analytics: Learning Analytics for Serious Games***. In: Learning, Design, and Technology. pp. 1–29. Springer International Publishing, Cham (2016).

(Serrano, 2017) Serrano-Laguna, Á., Martínez-Ortiz, I., Haag, J., Regan, D., Johnson, A., Fernández-Manjón, B.: ***Applying standards to systematize learning analytics in serious games***. Comput. Stand. Interfaces. 50, 116–123 (2017).